Práctica 3: Algoritmos Greedy

David Cabezas Berrido

Índice

0. Problema de las reparaciones **2**

1. Resolución teórica: Un único electricista **2**

1.1. Deducción del algoritmo 2

1.2. Demostración de que produce la solución óptima 2

1.2.1. Inducción sobre el número de reparaciones 2

1.2.2. Reducción al absurdo 3

0. Problema de las reparaciones

Un electricista necesita hacer n reparaciones, se sabe de antemano que la tarea i-ésima tardará t_i minutos. El objetivo es minimizar el tiempo medio de espera de los clientes. Debemos diseñar un algoritmo que tome una lista de tiempos de tareas como entrada y devuelva la permutación de la misma en la que el tiempo medio de espera de los clientes sea óptimo.

1. Resolución teórica: Un único electricista

1.1. Deducción del algoritmo

Las tareas se realizarán en el orden indicado por su subíndice: $[t_1, t_2, \ldots, t_n]$

Notaremos C_i al tiempo de espera del cliente i-ésimo. Este tendrá que esperar a que se realicen tanto su reparación como todas las anteriores, el primer cliente sólo tendrá que esperar el tiempo que tome su reparación ($C_1 = t_1$), el segundo el tiempo de la primera reparación y el de la suya ($C_2 = t_1 + t_2$). En general:

$$C_i = \sum_{j=1}^{i} t_j \qquad \forall i \in \{1, 2, \ldots, n\}$$

Nuestro objetivo es minimizar el tiempo medio de espera de cada cliente, por lo tanto la suma de los C_i debe ser lo más baja posible. Llamemos C a esta suma:

$$C = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \sum_{j=1}^{i} t_j = \sum_{i=1}^{n} (n + 1 - i) t_i$$

Como observamos, los tiempos de las tareas quedan multiplicados por coeficientes cada vez menores. Por tanto, el tiempo total de espera es menor cuando las reparaciones que requieren menos tiempo se realizan antes. Por ello, propongo como algoritmo realizar las tareas en orden de menor a mayor tiempo requerido.

1.2. Demostración de que produce la solución óptima

Para probar que este algoritmo produce la salida óptima, lo he hecho de dos formas: por inducción y por reducción al absurdo.

1.2.1. Inducción sobre el número de reparaciones

En el caso $n = 1$, hay una única solución y por tanto óptima.

Supongamos que nuestro algoritmo siempre encuentra la solución óptima en un problema de tamaño n, y consideremos un problema con una lista T de $n+1$ elementos con los tiempos de $n+1$ reparaciones urgentes. $T = [t_1, t_2, \ldots, t_n, t_{n+1}]$.

Notaremos $R = [r_1, r_2, \ldots, r_n, r_{n+1}]$ a la solución que produce nuestro algorítmo y $S = [s_1, s_2, \ldots, s_n, s_{n+1}]$ a una solución óptima (existe porque el número de posibles soluciones es finito, $n!$). Ambas son permutaciones de T. Tenemos:

$$C_r = \sum_{i=1}^{n+1} (n + 2 - i) r_i \qquad C_s = \sum_{i=1}^{n+1} (n + 2 - i) s_i$$

Sabemos que $C_s \leq C_r$, ya que S es la solución óptima. Debemos probar que se da la igualdad.

Tomemos el primer sumando de cada sumatoria

$$C_r = (n + 1) r_1 + C_r^* \qquad C_r^* = \sum_{i=2}^{n+1} (n + 2 - i) r_i$$

$$C_s = (n+1)s_1 + C_s^* \qquad C_s^* = \sum_{i=2}^{n+1}(n+2-i)s_i$$

Como nuestro algoritmo prioriza siempre la reparación que menos tiempo requiera, podemos asegurar $r_1 \leq s_1,\ (n+1)r_1 \leq (n+1)s_1$. Luego como sabíamos que $C_s \leq C_r$, deducimos que $C_s^* \leq C_r^*$.

Consideramos ahora los problemas dados por $R^* = [r_2, r_3, \ldots, r_n, r_{n+1}]$ y $S^* = [s_2, s_3, \ldots, s_n, s_{n+1}]$, ambos de tamaño n. Sabemos que nuestro algoritmo produce para R^* la solución óptima, con C_r^* como suma de los tiempos de espera ya que

$$\sum_{i=1}^{n}(n+1-i)r_{i+1} = \sum_{i=2}^{n+1}(n+2-i)r_i = C_r^*$$

Si $r_1 = s_1$, R^* y S^* serían el mismo problema, por lo que C_s^* sería una suma de tiempos para alguna solución del problema dado por R^*, para el que C_r^* es mínima. Pero sabíamos previamente que $C_s^* \leq C_r^*$, luego se dará la igualdad como queríamos.

$$C_s^* = C_r^*,\ (n+1)r_1 = (n+1)s_1 \implies C_s = C_r$$

Probemos que esta es la única posibilidad llegando a un absurdo partiendo de $r_1 \neq s_1$.

En este caso tendremos $r_1 < s_1$, extraemos el tiempo r_1 de la lista S, supondremos que se corresponde con s_k con $k > 1$. Tendremos entonces

$$\begin{gathered}
s_1 > r_1 = s_k \qquad n+2-k < n+1 \\
s_1 > s_k \qquad n+1-(n+2-k) > 0 \\
(n+1-(n+2-k))s_1 > (n+1-(n+2-k))s_k \\
(n+1)s_1 - (n+2-k)s_1 > (n+1)s_k - (n+2-k)s_k
\end{gathered}$$

y por tanto

$$(n+1)s_1 + (n+2-k)s_k > (n+1)s_k + (n+2-k)s_1$$

Esto significa que intercambiar las posiciones de los tiempos s_1 y s_k en la lista S produciría una suma de tiempos menor a la existente, lo que contradice la hipótesis de que S es una solución óptima.

□

1.2.2. Reducción al absurdo

Supongamos que un problema de reparaciones dado por la lista $T = [t_1, t_2, \ldots, t_n]$, sean $R = [r_1, r_2 \ldots, r_n]$ y $S = [s_1, s_2 \ldots, s_n]$ dos permutaciones de T. Donde R es la salida que produciría nuestro algoritmo y S es la solución óptima. Sabemos que R cumple $r_1 \leq r_2 \leq \ldots \leq r_n$, probemos que S debe cumplir lo mismo.

La suma de los tiempos de espera de S es menor o igual que la de cualquier otra solución. Por tanto $C_s = \sum_{i=1}^{n}(n+1-i)s_i$ es mínima. Supongamos que S no estuviera ordenada en orden no decreciente, esto es

$$\exists p, q \in \{1, 2, \ldots, n\} \text{ tales que } p < q \text{ y } s_p > s_q$$

Tendremos entonces

$$\begin{gathered}
s_p > s_q \qquad n+1-q > n+1-p \\
s_p > s_q \qquad ((n+1-q)-(n+q-p)) > 0 \\
((n+1-q)-(n+q-p))s_p > ((n+1-q)-(n+q-p))s_q \\
(n+1-q)s_p - (n+1-p)s_p > (n+1-q)s_q - (n+1-p)s_q
\end{gathered}$$

y por tanto

$$(n+1-q)s_p + (n+1-p)s_q > (n+1-q)s_q + (n+1-p)s_p$$

Esto significa que intercambiar las posiciones de los tiempos s_p y s_q en la lista S produciría una suma de tiempos menor a la existente, lo que contradice la hipótesis de que S es una solución óptima. Tenemos entonces $s_1 \leq s_2 \leq \ldots \leq s_n$, luego $S = R$ como queríamos.

□